Theoretical Computer Science 56 (1988) 37-57
North-Holland

REWRITING WITH A NONDETERMINISTIC CHOICE OPERATOR

Stéphane KAPLAN*
LRI, Bâtiment 490, Université des Sciences, F-91405 Orsay Cedex, France

Abstract. The privileged field of classical algebra and term-rewriting systems is that of strictly deterministic systems: the *confluence* property is generally assumed to hold, which implies determinism for the result of the computations, even if there exist several different computation paths. In this paper, we introduce a bounded nondeterministic choice operator "$\cup$" into algebraic specifications and related term-rewriting systems. The operator "$\cup$", which constructs *sets of values*, satisfies AC (associative-commutative) properties, which allows to apply results about equational rewriting. Attention is then mainly restricted to so-called regular systems, where nondeterministic choice is constraint-free. Several examples are considered, including a toy concurrent language, for which nontrivial properties may be automatically proved.

Key words. Term-rewriting systems, confluence, nondeterminism, equational rewriting.

Introduction

The field of term-rewriting systems has greatly developed during the past several years. It provides an appropriate operational description for algebraic specifications, defining abstract implementation by means of symbolic evaluation within the term algebra. Several large theorem-proving systems have been developed along these lines. However, an essential constraint on those systems is that they must be *confluent.* Roughly speaking, the confluence property means that, even if different computational paths are possible in order to evaluate a given term, the result must be unique. Therefore, it has not been possible so far to treat nondeterministic and, in particular, concurrent specifications in that framework, in a fully satisfying way.

In this paper, we propose a new formalism allowing to introduce explicit nondeterminism in term-rewriting systems. This formalism extends the classical (strictly deterministic) one, while maintaining its most important properties: coherence between the algebraic and the operational aspects, possibilities of automatic theorem proving, etc. Our main idea is to introduce a special purpose operator "$\cup$", which realizes bounded nondeterministic choice by constructing sets of results. For instance, the fact that a constant "a" may rewrite into "b" or "c" is specified in our formalism by the rule: $a \to b \cup c$. All the nonconfluence that is permitted in such systems is compelled to derive from the sole operator "$\cup$".

* Present affiliation: Leibniz Center for Research in Computer Science, Institute of Mathematics and Computer Science, Hebrew University, Gavat-Ram, 91904 Jerusalem, Israel.

In a first approach to this question [29], we defined a new algebraic formalism, together with its associated rewriting counterpart. It appears that slightly restricting the kind of nondeterminism that is allowed, the "$\cup$" operator may be modeled in a *classical* algebraic framework, that is in turn associated with term-rewriting systems *modulo* associative-commutative properties of "$\cup$"; this paper presents the latter approach.

Our formalism can be used to deal with concurrent specifications, much in the flavor of the *process algebras* of the ACP group (cf. [4, 5, 6]). The kernel of a concurrent language is presented to illustrate this aspect. Moreover, the approach developed here is at the basis of our work on the algebraic specification of concurrently accessed data structures [31].

A crucial hypothesis about this formalism is that all the *computations eventually terminate*, as it is the case for classical term-rewriting systems. Thus, we need not consider the properties of infinite nondeterministic calculi, as found, for instance in [12, 20, 40, 43].

Section 1 defines the algebraic basis of our approach, via the notion of $\cup$-specifications. Section 2 introduces $\cup$-term-rewriting systems, with their interpretation as equational systems. In Section 3, we restrict our attention to the class of *regular* systems; results about confluence are discussed. Lastly, Section 4 illustrates the application of our methods to the proof of inductive properties.

We assume that the reader has a basic knowledge of algebras and term-rewriting systems (cf. [1, 24] for basic references), nondeterminism (cf. [2, 40]) and concurrency (cf. [9, 10, 25, 38]). However, notations and concepts are systematically redefined, and the paper is intended to be self-contained.

1. The algebraic framework

In this section, we first recall classical notions about algebraic specifications and term algebras. We then introduce our notion of $\cup$-specification.

1.1. Classical specifications

We briefly recall several definitions, that are usual in the fields of abstract data types and term-rewriting systems (cf. [1, 24] for further details).

A *signature* consists of:

- a set S of domain identifiers, called *sorts*.
- a set Σ of function identifiers, with an arity function $\mathrm{ar}: \Sigma \to S^+$. If $\mathrm{ar}(f) = s_1 \ldots s_n s$, we write $f: s_1 \times \cdots \times s_n \to s$.

We denote by $T_{S,\Sigma}$ the set of all terms built as usual on (S, Σ) (often called ground terms). $T^s_{S,\Sigma}$ stands for the set of ground terms of sort s. Let $X = (X_s)_{s \in S}$ be an infinite set of typed variables; $T_{S,\Sigma}(X)$ denotes the set of terms with variables. For $t \in T_{S,\Sigma}(X)$, $\mathrm{Var}(t)$ stands for the variables occurring in t. *Substitutions* are defined in the usual way. The application of a substitution σ to a term t is written

$t\sigma$. A *context* is a term $K[X_s]$ of $T_{S,\Sigma}(X)$ with a distinguished occurrence of a variable X_s of type $s \in S$. An *equation* on signature (S, Σ) is a pair (t, t') of terms with variables belonging to the same sort; it is usually written $t = t'$.

An *algebraic specification* is a triple SPEC $= \langle S, \Sigma, E \rangle$, where (S, Σ) is a signature, and E a set of equations on this signature.

A *model* M of specification SPEC consists of:

- for each $s \in S$, a domain M^s,
- for each $f \in \Sigma$, with $f: s_1 \times \cdots \times s_n \to s$, an application $f^M : M^{s_1} \times \cdots \times M^{s_n} \to M^s$ such that, for any equation $t = t'$ of E and for any ground substitution $\sigma: X \to T_{S,\Sigma}$, we have

$$\text{eval}_M(t\sigma) = \text{eval}_M(t'\sigma) \qquad (*)$$

where $\text{eval}_M : T_{S,\Sigma} \to M$ is recursively defined by $\text{eval}_M(f(t_1, \ldots, t_n)) = f^M(\text{eval}_M(t_1), \ldots, \text{eval}_M(t_n))$. Property $(*)$ is usually written: $M \models t = t'$.

Given two models M and M' of a specification SPEC, a *morphism* ϕ from M to M' is a family of applications $\varphi_s : M^s \to M'^s$ such that, for any $f: s_1 \times \cdots \times s_n \to s$, and for any $m_i \in M^{s_i}$

$$\varphi_s(f^M(m_1, \ldots, m_n)) = f^{M'}(\varphi_{s_1}(m_1), \ldots, \varphi_{s_n}(m_n)).$$

Mod_{SPEC}, the set of all the models of a specification SPEC with its morphisms, is a nonempty category.

A *congruence* $\equiv$ is a family $(\equiv_s)_{s \in S}$ of equivalence relations on $T_{S,\Sigma}$ (respectively $T_{S,\Sigma}(X)$) such that, for any $f: s_1 \times \cdots \times s_n \to s$ and for any $t_i, t_i' \in T_{S,\Sigma}(X)^{s_i}$ (respectively $T_{S,\Sigma}^{s_i}$), if $\forall i \in [1, \ldots, n]$, $t_i \equiv_{s_i} t_i'$, then $f(t_1, \ldots, t_n) \equiv_s f(t_1', \ldots, t_n')$. Moreover, if $\equiv$ is on $T_{S,\Sigma}(X)$, it should be the case that if $t \equiv t'$, then $t\sigma \equiv t'\sigma$, for any substitution σ. A set of equations E being given, there exists a smallest congruence $\equiv_E$ on $T_{S,\Sigma}$ that contains all the couples $(K[M\sigma], K[M'\sigma])$, where $M = M'$ ranges over E, $K[X]$ is a context, and σ a ground substitution. Then, $T_{S,\Sigma}/\equiv_E$ is the *initial model* of SPEC.

We consider *occurrences* in terms as finite strings of integers in the usual manner. For a term t and an occurrence ω in t, $t_{|\omega}$ stands for the subterm of t the root of which is at occurrence ω. $t[\omega \leftarrow t']$ is the term t, where $t_{|\omega}$ is replaced by the term t'. It is understood in this case that the sort of t' is also the sort of $t_{|\omega}$.

We say that a term t *matches* the term λ, called a pattern, at occurrence ω via the substitution σ if $t_{|\omega} = \lambda\sigma$. Two terms t and t' are *unifiable* if there exists a substitution σ such that $t\sigma = t'\sigma$. In that case, they admit a most general unifier, i.e., a substitution μ that is a unifier of t and t', and such that, for every unifier σ of t and t', there exists a substitution σ' such that $\sigma = \mu\sigma'$.

Let E be a set of equations on $T_{S,\Sigma}(X)$. We say that t and t' are *E-unifiable* if there exists a substitution σ such that $t\sigma \equiv_E t'\sigma$. A *complete* set of E-unifiers of t and t' is a set C of E-unifiers of t and t' such that, for any E-unifier ρ of t and t', there exists a $\sigma \in C$ and a substitution τ such that $\rho = \sigma\tau$.

For a given relation $\rightarrow$, $\rightarrow^*$ denotes its reflexive and transitive closure, and $\rightarrow^{-1}$ its inverse. Thus, $(\rightarrow \cup \rightarrow^{-1})^*$ denotes the reflexive, symmetric and transitive closure of $\rightarrow$.

1.2. $\cup$-Specifications

We now present our specific notion of specification with explicit nondeterministic operator.

Definition 1.1. A *$\cup$-specification* is a classical specification $\langle S, \Sigma, E^{\text{def}} + E_\cup \rangle$ such that:

(1) for a subset $S^\cup$ of S, for any $s \in S^\cup$, there exist two operators:

$$\emptyset_s : \rightarrow s \quad \text{and} \quad \cup_s : s \times s \rightarrow s;$$

(2) for every $f : s_1 \times \cdots \times s_n \rightarrow s$, if $s_i \in S^\cup$ for some $i \in [1, \ldots, n]$, then $s \in S^\cup$;

(3) $E_\cup$ is the following set of equations: for any $s \in S^\cup$,

$$\forall_s x, y, z \quad (x \cup_s y) \cup_s z = x \cup_s (y \cup_s z), \tag{A}$$

$$\forall_s x, y \quad x \cup_s y = y \cup_s x, \tag{C}$$

$$\forall_s x \quad x \cup_s x = x, \tag{I}$$

$$\forall_s x \quad x \cup_s \emptyset_s = x; \tag{N}$$

for any $f : s_1 \times \cdots \times s_n \rightarrow s$ with $s_i \in S^\cup$, $\forall_{s_j} x_j (j \neq i)$, $\forall_{s_i} x, y$,

$$f(x_1, \ldots, x_{i-1}, x \cup_{s_i} y, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) \cup_s f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n). \tag{D}$$

The intuition is that the sorts of $S^\cup$ represent *sets of values*; for $s \in S^\cup$, $\cup_s$ represents the *union* of sets of values of sort s, and $\emptyset_s$ is the empty set of values of sort s. This statement is made precise in Section 3 (Lemma 3.2 and Theorem 3.3). Property (2) then states that if an operator may take a set of values as one of its arguments, it has to be able to return a set of values. Lastly, the equations of $E_\cup$ respectively state the properties of associativity, commutativity, idempotence, neutrality and distributivity of the $\cup_s$ and $\emptyset_s$ (abbreviated from now on by the letters "A", "C", "I", "N" and "D").

The equations of E^{def} are those defined by the users. It is the part (together with S and Σ) that allows to specify different $\cup$-specifications.

We sometimes use the concept of *m$\cup$-specification*, which is identical to a $\cup$-specification except that equation (I) need not hold. Thus, operators into $S^\cup$ will produce *multisets* of results, as made explicit in Section 3.

Example 1.2. We consider the kernel, a small programming language with concurrency, presented in [29]. Concurrency is modeled as in the formalism of the *process algebras* (cf. [4, 5, 6]). In this language, processes apply to values (and produce sets

of values) via an application operator "::" *à la Backus.* Its $\cup$-specification is as follows:

- $S = \{\text{values}, \text{process}\}$, $S^{\cup} = \{\text{values}\}$;
- $\Sigma = \{a, b, \ldots : \to \text{process}\}$,

  $+ : \text{process} \times \text{process} \to \text{process}$,

  $; : \text{process} \times \text{process} \to \text{process}$,

  $\| : \text{process} \times \text{process} \to \text{process}$,

  $\lfloor\!\lfloor : \text{process} \times \text{process} \to \text{process}$,

  $:: : \text{process} \times \text{values} \to \text{values}\}$;
- $E^{\text{def}} = \{\forall_{\text{process}} p, q, r, \quad (p;q);r = p;(q;r),$

  $\forall_{\text{process}} p, q, \quad p \| q = p \lfloor\!\lfloor q + q \lfloor\!\lfloor p,$

  $\forall_{\text{process}} p, q, \quad (a;p) \lfloor\!\lfloor q = a;(p \| q),$

  $(b;p) \lfloor\!\lfloor q = b;(p \| q), \ldots,$

  $\forall_{\text{process}} p, q, \forall_{\text{values}} x, \quad (p+q)::x = (p::x) \cup (q::x),$

  $\forall_{\text{process}} p, q, \forall_{\text{values}} x, \quad (p;q)::x = (p::(q::x))\}.$

For instance, we have

$$(a;b \| a;c)::x \equiv (a;a;b;c)::x \cup (a;a;c;b)::x \cup (a;b;a;c)::x \cup (a;c;a;b)::x.$$

In a given instance of the language, to terms such as "$a::x$" a particular meaning would be assigned, which would describe the language in a complete fashion (cf. [29]). In Section 4, we will present proofs for several properties of this language.

Example 1.3. Let us consider a Petri net N (cf., e.g., [44, 45], from which we borrow the notations). We suppose that the transitions of N may be passed in either directions (this condition is dropped later, when considering $\cup$-term-rewriting systems that are naturally oriented). We associate to N the m$\cup$-specification $\bar{N}$ defined by

- $S_{\bar{N}} = S^{\cup} = \{\text{places}\}$;
- $\Sigma_{\bar{N}}$ contains only constants of sort "places", exactly one per place of the net;
- let τ be a transition of the net, let $p_1, \ldots, p_n$ be the transitions leading to τ, and $q_1, \ldots, q_n$ the transitions issued from τ: by (e_τ) we define the equation:

  $$q_1 \cup \cdots \cup q_n = q'_1 \cup \cdots \cup q'_m$$

  and we let $E_{\bar{N}}^{\text{def}} = \{(e_\tau) \mid \tau \in T_N\}$.

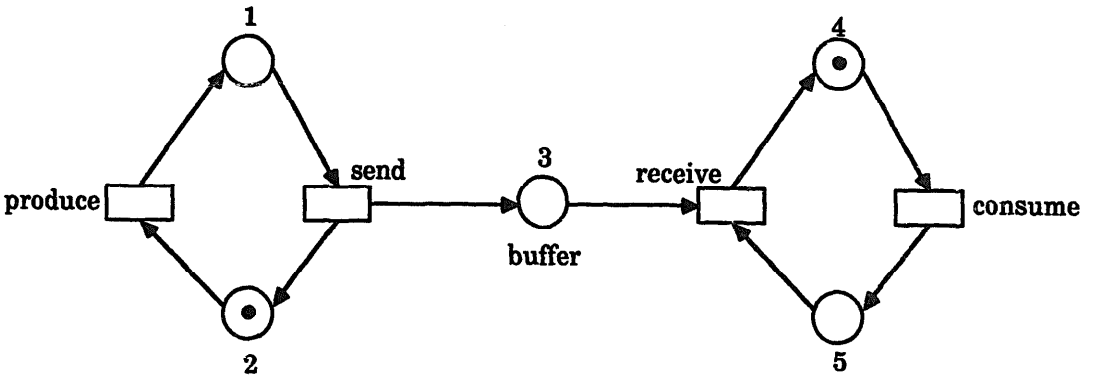

Fig. 1.

For instance, to the net shown in Fig. 1, the following equations are associated:

$$E_N^{\text{def}} = \{e_{\text{produce}}: \ p_2 = p_1,$$
$$e_{\text{send}}: \ p_1 = p_2 \cup p_3,$$
$$e_{\text{receive}}: \ p_3 \cup p_4 = p_5,$$
$$e_{\text{consume}}: \ p_5 = p_4\}.$$

Now, to a marking μ of the net, with k_i tokens in the place q_i (for $i \in [1, \ldots, n]$), we associate the term

$$\vartheta_\mu = q_1^{k_1} \cup \cdots \cup q_n^{k_n}$$

with $q^1 = q$ and $q^{k+1} = q \cup q^k$. It is then easy to check that μ' is *accessible* from μ if and only if $\vartheta_\mu \equiv_{\bar{N}} \vartheta_{\mu'}$.

Conversely, to any m$\cup$-specification SPEC such that $S^{\cup} = S$, one may associate a Petri net π(SPEC) in the following way:

- the places of π(SPEC) are the ground terms containing no "$\cup$" or "$\emptyset$" symbols;
- to any ground instance of an equation of E^{def} *in flattened form* (cf. Section 2) $t_1 \cup \cdots \cup t_n \equiv_{\text{SPEC}} t'_1 \cup \cdots \cup t'_n$, is associated a (nonoriented) transition with arrows from $t_1, \ldots, t_n$ and to $t'_1, \ldots, t'_n$.

As before, to any term t in flattened from $t_1^{k_1} \cup \cdots \cup t_n^{k_n}$, we associate the marking of the net $\nu(t)$ with k_i tokens in the place t_i ($i \in [1, \ldots, n]$). Then $t \equiv_{\text{SPEC}} t'$ if and only if $\nu(t)$ is accessible from $\nu(t')$.

Moreover, for any Petri net N, $\pi(\bar{N}) = N$ and for any marking μ of N, $\nu(\vartheta_\mu) = \mu$. Similarly, for any m$\cup$-specification SPEC, $\overline{\pi(\text{SPEC})}$ is isomorphic to SPEC (i.e., the models of both specifications are the same) and, for any term t in flattened form, $\vartheta_{\nu(t)} = t$.

2. $\cup$-Term-rewriting systems

In this section, we present our notion of $\cup$-term-rewriting systems, which is the natural operational interpretation of $\cup$-specifications. Since these systems are

equational, we first recall essential properties of equational term-rewriting systems (cf. [26, 27, 28, 34]).

2.1. General equational term-rewriting systems

We suppose that a set of equations E and a set of rewrite rules R are given on $T_{S,\Sigma}(X)$. We define several notions of rewriting on $T_{S,\Sigma}(X)$.

Definition 2.1. Given two terms t and t' in $T_{S,\Sigma}(X)$, we have

- $t \to_{R/E} t'$ iff there exist u and u' in $T_{S,\Sigma}(X)$ such that $t \equiv_E u$, $u \to_R u'$ and $u' \equiv_E u$;
- $t \to_{R,E} t'$ iff there exist a rule $\lambda \to \rho$ in R, an occurrence ω of t, and a substitution σ such that $t_{|\omega} \equiv_E \lambda\sigma$ and $t' = t[\omega \leftarrow \rho\sigma]$.

We suppose that $R = L + \mathrm{NL}$, where the rules of L are *left-linear.* Then,

- $t \to_{L \cup \mathrm{NL}E} t'$ iff
  - *either* there exist a rule $\lambda \to \rho$ in L, an occurrence ω of t, and a substitution σ such that $t_{|\omega} = \lambda\sigma$ and $t' = t[\omega \leftarrow \rho\sigma]$,
  - *or* there exist a rule $\lambda \to \rho$ in NL, an occurrence ω of t, and a substitution σ such that $t_{|\omega} \equiv_E \lambda\sigma$ and $t' = t[\omega \leftarrow \rho\sigma]$.

$\to_{R/E}$ simulates the rewriting on the classes modulo E. However, it requires *two* steps of E-equality, which is costly. This justifies the consideration of the other rewriting relations; Results 2.4 and 2.5 hereafter then ensure that, under good hypotheses, all these rewritings convey the same information. $\to_{R,E}$ has been introduced by Peterson and Stickel [42], and $\to_{L \cup \mathrm{NL}E}$ is considered, for instance, in [27, 34]. Note that we have

$$\to_{R/E} \supseteq \to_{R,E} \supseteq \to_{L \cup \mathrm{NL}E} \supseteq \to_R.$$

Definition 2.2. A finite set of equations E is *smooth* if

- E-pattern matching is decidable;
- E-unification is decidable, and there exists an algorithm computing a minimal set of E-unifiers of two given terms;
- $\equiv_E$ is decidable;
- the congruence classes of $\equiv_E$ are finite.

When E is smooth, the four rewriting relations considered above are decidable. Note that these conditions are satisfied in particular when $E = \mathrm{AC}$ (i.e., associativity and commutativity of the $\cup_s$'s), which is the case in this paper.

Definition 2.3. A binary relation $\to$ on $T_{S,\Sigma}(X)$ is *E-terminating* iff there exist no infinite chain:

$$t_1 \equiv_E t_2 \to t_3 \equiv_E t_4 \to \cdots \to t_n \equiv_E t_{n+1} \to \cdots.$$

Result 2.4. *Let $\to$ be a binary relation on $T_{S,\Sigma}(X)$ such that $\to_{R/E} \supseteq \to \supseteq \to_R$. The following properties are equivalent:*

(i) *$\to_{R/E}$ is R-terminating,*
(ii) *$\to$ is E-terminating,*
(iii) *$\to_R$ is E-terminating.*

Definition and property 2.5. Let E be a finite set of equations. A binary relation $\rightarrow$ on $T_{S,\Sigma}(X)$ is *E-Church-Rosser* iff, for any terms t, t' of $T_{S,\Sigma}(X)$ such that $t\,(\equiv_E \cup \rightarrow \cup \rightarrow^{-1})^* t'$, there exist two terms u, u' such that $t \rightarrow^* u$, $t' \rightarrow^* u'$, $u \equiv_E u'$.

If $\rightarrow$ is E-Church-Rosser and E-terminating, then, for any terms t, t' of $T_{S,\Sigma}(X)$ we have

$$t\,(\equiv_E \cup \rightarrow \cup \rightarrow^{-1})^* t' \text{ iff } t{\downarrow} \equiv_E t'{\downarrow},$$

where $t{\downarrow}$ and $t'{\downarrow}$ are any normal form of t and t' for $\rightarrow$.

This states that, under the condition that $\rightarrow$ is E-terminating and E-Church-Rosser, it is sufficient to consider the normal form of two terms in order to decide whether they are in relation via $(\equiv_E \cup \rightarrow \cup \rightarrow^{-1})^*$.

Result 2.5 typically applies to $\rightarrow$ being one of the rewriting relations defined above (Definition 2.1). Note that, in this case, and more generally for any $\rightarrow$ such that $\rightarrow_{R/E} \supseteq \rightarrow \supseteq \rightarrow_R$, we have

$$(\equiv_E \cup \rightarrow \cup \rightarrow^{-1})^* = \equiv_{E+R}.$$

In this last expression, R is considered as a set of equations. Also, Result 2.4 holds when replacing "E-terminating" by "E-Church-Rosser". Thus, it stems from Property 2.5 that deciding the equality of two terms modulo $E+R$ simply amounts to checking whether their normal forms for any $\rightarrow$ between $\rightarrow_{R/E}$ and $\rightarrow_R$ are equal modulo E.

Lastly, we recall results *à la Knuth-Bendix* allowing to decide whether the previous rewriting relations are E-Church-Rosser, knowing that they are E-terminating.

Definition 2.6. Let R and R' be two sets of rules. We define:

(1) $\mathrm{CP}(R, R')$, the set of *critical pairs* of R against R', as the set of pairs $\langle t, t'\rangle$ such that:

- there exist a rule $\lambda \rightarrow \rho$, a rule $\lambda' \rightarrow \rho'$ in R', a nonvariable occurrence ω in λ such that $\lambda_{|\omega}$ and λ' are unifiable, of m.g.u. σ,
- $t = \rho\sigma$ and $t' = \lambda\sigma[\omega \leftarrow \rho'\sigma]$;

(2) $\mathrm{ECP}(R, R')$, the set of *E-critical pairs* of R against R', as the set of pairs $\langle t, t'\rangle$ such that

- there exist a rule $\lambda \rightarrow \rho$, a rule $\lambda' \rightarrow \rho'$ in R', a nonvariable occurrence ω in λ such that $\lambda_{|\omega}$ and λ' are E-unifiable, with minimal set of E-unifiers Λ,
- $t = \rho\sigma$ and $t' = \lambda\sigma[\omega \leftarrow \rho'\sigma]$, where σ ranges over Λ.

Let also $\mathrm{CP}(E, R) = \mathrm{CP}(E^{\text{left-to-right}}, R) + \mathrm{CP}(E^{\text{right-to-left}}, R)$ where $E^{\text{left-to-right}}$ stands for the equations of E oriented from left to right, etc. Note that, under our assumptions on E, all the previous sets of critical pairs are computable.

We say that a critical pair $\langle t, t'\rangle$ is *convergent* iff $t{\downarrow} \equiv_E t'{\downarrow}$. A set of critical pairs is convergent when each of its pairs is convergent. We then have the following main theorem (cf., e.g., [27]).

Theorem 2.7. *Let E be a smooth set of equations on $T_{S,\Sigma}(X)$. Let R be a set of rules on $T_{S,\Sigma}(X)$ such that $R = L + \mathrm{NL}$, where L contains only left-linear rules. We suppose that R is E-terminating. Then $\to_{L \cup \mathrm{NLE}}$ is E-Church-Rosser iff*

(1) *any critical pair in* $\mathrm{CP}(L, L) + \mathrm{CP}(L, \mathrm{NL}) + \mathrm{CP}(L, E) + \mathrm{CP}(E, L)$ *is convergent for* $\to_{L \cup \mathrm{NLE}}$, *and*

(2) *any E-critical pair in* $\mathrm{ECP}(\mathrm{NL}, \mathrm{NL}) + \mathrm{ECP}(\mathrm{NL}, L) + \mathrm{ECP}(\mathrm{NL}, E)$ *is convergent for* $\to_{L \cup \mathrm{NLE}}$.

This gives a decision procedure for $\to_{L \cup \mathrm{NLE}}$ being E-Church-Rosser (and thus for any $\to$ between $\to_{R/E}$ and $\to_R$) under the hypotheses of the theorem. Notice that in the case where $\mathrm{NL} = \emptyset$, we obtain the theorem of [22], stating that it is enough to check for the convergence of $\mathrm{CP}(L, L) + \mathrm{CP}(L, E) + \mathrm{CP}(E, L)$.

2.2. $\cup$-Term-rewriting systems

Definition 2.8. A *$\cup$-term-rewriting systems* (abbreviated into $\cup$-TRS) is an equational term-rewriting system (R, E) on the signature of a $\cup$-specification, where:

(1) E is the following set of equations: for any $s \in S^{\cup}$,

$$\forall_s x, y, z \quad (x \cup_s y) \cup_s z = x \cup_s (y \cup_s z), \tag{A}$$

$$\forall_s x, y \quad x \cup_s y = y \cup_s x. \tag{C}$$

(2) $R = L + \mathrm{NL}$, and:

- $L = R_{\mathrm{N}} + R_{\mathrm{D}} + R_{\mathrm{L}}^{\mathrm{def}}$, with

$$\text{for any } s \in S^{\cup}, \quad x \cup_s \emptyset_s \to x, \tag{R_{N}}$$

$$\text{for any } f: s_1 \times \cdots \times s_n \to s \text{ with } s_i \in S^{\cup},$$

$$f(x_1, \ldots, x_{i-1}, x \cup_{s_i} y, x_{i+1}, \ldots, x_n)$$

$$\to f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) \cup_s f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n); \tag{R_{D}}$$

- $\mathrm{NL} = R_{\mathrm{I}} + R_{\mathrm{NL}}^{\mathrm{def}}$, with

$$\text{for any } s \in S^{\cup}, \quad x \cup_s x \to x; \tag{R_{I}}$$

- $R_{\mathrm{L}}^{\mathrm{def}}$ and $R_{\mathrm{NL}}^{\mathrm{def}}$ are user-defined set of rules, such that $R_{\mathrm{L}}^{\mathrm{def}}$ is left-linear.

So, with respect to $\cup$-specifications of Section 2, the equations of $E_{\cup}$ are simulated by E, R_{I}, R_{N} and R_{D}, while E^{def} is simulated by $R_{\mathrm{L}}^{\mathrm{def}}$ and $R_{\mathrm{NL}}^{\mathrm{def}}$.

Other ways of splitting the equations might have been considered. We have chosen to let $E = \mathrm{AC}$ in particular, because there exist rewrite rule laboratories, such as the REVE system [34, 37], that allow to experiment with such kind of rules. Actually, it has been possible to validate the approach suggested in this paper through several experiments under REVE-2, as reported hereafter, simulating with success our notion of rewriting.

Notice however that the "I" equations could not be integrated into E since the congruence classes of $\equiv_E$ would not be finite any longer in this case. Also, and to our knowledge, matching and unification modulo AC and "D" have not been studied systematically so far (cf. [33] for general reference on the topic).

As before, we will sometimes refer to m$\cup$-TRS, which are $\cup$-TRS without the rewrite rules "R_I". For instance, let us consider again Petri nets, as in Example 1.3, but assuming now that the transitions are *oriented* (as they actually are). Equation (e_τ) is now interpreted by the rule (r_τ):

$$q_1 \cup \cdots \cup q_n \to q'_1 \cup \cdots \cup q'_m$$

which is in R_{NL}^{def} (or possibly in R_L^{def}). Then, with the same conventions, a marking μ' of the net is accessible from a marking μ iff $\vartheta_\mu \to_{R/E} \vartheta_{\mu'}$. Several properties of Petri nets may be studied in this fashion (cf. also [14] for related approach, using nonequational systems). Notice however that the hypothesis of E-termination is equivalent to the fact that all the computations starting from any initial marking terminate, which is a strong requirement.

2.3. *Properties of basic $\cup$-TRS*

Here we consider properties of the $\cup$-system such that $R_L^{def} = R_{NL}^{def} = \emptyset$. We have $E = \mathrm{AC}$ and $R = \mathrm{IND}$. It may be considered as the *basic* system that is contained in any $\cup$-system. In this case, $\to_{L \cup NLE}$ is simply $\to_{R,E} = \to_{\mathrm{IND},E}$. We have the following result:

Theorem 2.9. (1) *The relation $\to_{\mathrm{IND},E}$ is E-terminating and E-Church–Rosser. The E-normal form of a term for $\to_{\mathrm{IND},E}$ is called its* quasi-flattened *form.*

(2) *Let $\to_0$ stand for $\to_{\mathrm{IND+(Flat)},E}$ where* (Flat) *is the family of rules*

$$\bigcup_s (t_1, \ldots, t_n, \cup_s (t_{n+1}, \ldots, t_{n+m}), t_{n+m+1}, \ldots, t_{n+m+p})$$
$$\to \bigcup_s (t_1, \ldots, t_n, t_{n+1}, \ldots, t_{n+m}, t_{n+m+1}, \ldots, t_{n+m+p}),$$

"s" being a sort of $S^\cup$ and the $\cup_s$ being considered as associative-commutative operators of variable arity. Then $\to_0$ is E-terminating and E-Church–Rosser. The normal form for $\to_0$ of a term t of sort $s \in S^\cup$ is called its flattened form, *and denoted by* flat(t).

For instance, if $t = f(a \cup b \cup a, g(c \cup \emptyset, d \cup e))$, its flattened form (modulo E) is

$$f(a, g(c, d)) \cup f(a, g(c, e)) \cup f(b, g(c, d)) \cup f(b, g(c, e)).$$

The proof of Theorem 2.9 is obtained via Theorem 2.7. See also [28]. It implies in particular that

$$t \equiv_{\mathrm{ACIND}} t' \text{ iff } \mathrm{flat}(t) \equiv_E \mathrm{flat}(t').$$

In the following, we will generally assume that the terms under consideration are in flattened or quasi-flattened form, as well as the left-hand sides and the right-hand sides of the rules.

3. Regular systems and confluence results

In this section, we restrict attention to *regular* systems, defined hereafter, which share interesting properties. Note that Theorem 2.7 provides means of checking whether *any* E-terminating $\cup$-TRS (not necessarily regular) is E-Church-Rosser, by computation of the several sets of critical pairs. However, we are going to give a refined version of this result adapted to the case of regular systems.

Definition 3.1. (1) A $\cup$-TRS is *left-$\cup$-free* if the left-hand side of each rule of $R_{\mathrm{L}}^{\mathrm{def}}$ and $R_{\mathrm{NL}}^{\mathrm{def}}$ contains no "$\cup_s$" or "$\emptyset_s$" symbol.

(2) A $\cup$-TRS is *regular* if it is left-$\cup$-free and $R_{\mathrm{NL}}^{\mathrm{def}}$ is empty.

Similarly, we say that a *term* is $\cup$-free if it contains no "$\cup_s$" or "$\emptyset_s$" symbol.

The condition of left-$\cup$-freeness somehow states that the nondeterministic choice is *context-free*: the set of the results of a computation represented by $\mathrm{exp}_1 \cup \mathrm{exp}_2$ is exactly the union of the set of the results of exp_1 with the set of the results of exp_2. This condition is for instance satisfied with the language presented in Example 1.2, and to our opinion characterizes methodologically the nondeterministic choice operators.

On the other hand, Petri nets representations are not in general left-$\cup$-free (for instance, the one considered in Example 1.3 is not). Indeed, in this case, the "$\cup$" operator represents synchronization ("$p \cup q$" is read: there is a token in p *and* a token in q), and not nondeterministic choice.

Note lastly that the condition of left-linearity is of technical order, as appears later on. We remark that for a regular system, $\rightarrow_{L \cup \mathrm{NL}E}$ is simply reduced to $\rightarrow_{R,E}$.

From now on, we suppose that a regular $\cup$-system is given, which is E-terminating and E-Church-Rosser. Note that for any $t \in T_{S,\Sigma}(X)^s$, with $s \in S^{\cup}$, the normal form of t (modulo AC) is of the form: $t_1 \cup_s \cdots \cup_s t_n$,[1] where the t_i's are *distinct* $\cup$-free terms of $T_{S,\Sigma}(X)^s$ in normal form; this derives from the fact that $\rightarrow_{R,E}$ contains $\rightarrow_{\mathrm{IND},E}$ (cf. Section 2.3). We have the following lemma.

Lemma 3.2. *For any $t, t' \in T_{S,\Sigma}(X)^s$ with associated normal form $t_1 \cup_s \cdots \cup_s t_n$ and $t'_1 \cup_2 \cdots \cup_s t'_m$, we have that $t''_1 \cup_s \cdots \cup_s t''_p$ is the normal form of $t \cup_s t'$, where the set $\{t''_1, \ldots, t''_p\}$ is the* set-theoretic union *of the sets $\{t_1, \ldots, t_n\}$ and $\{t'_1, \ldots, t'_m\}$.*

We now provide a characterization of the initial model of the $\cup$-specification associated to a regular system, in terms of *sets of values.* This allows to justify the intuition about the interpetation of the "$\cup$" and "$\emptyset$" symbols.

To this effect, let (R, E) be a given $\cup$-TRS on a signature (S, Σ), and let $s \in S^{\cup}$. We define:

- $A^s = \{t \in T_{S,\Sigma}(X)^s \mid t$ is a $\cup$-normal form$\}$,
- PA^s as the set of the finite subsets of A^s.

[1] This expression conventionally being "$\emptyset_s$" if $n = 0$, and "t_1" if $n = 1$.

We now define a (S, Σ)-algebra PA in the following fashion:

- if $s \in S^{\cup}$, PA^s is the set of all the normal forms of sort s;
- if $s \in S^{\cup}$, $\emptyset_s^{\mathrm{PA}}$ is the empty set;
- if $s \in S^{\cup}$, $\cup_s^{\mathrm{PA}}$ is the set-theoretic *union* in PA^s;
- for $f: s_1 \times \cdots \times s_n \to s$ such that $s \in S^{\cup}$ and if $s_i \in S^{\cup}$ and $s_j \notin S^{\cup}$ for $j \neq i$, then, if the terms $(t_j)_{j \neq i}$ are normal forms,

$$f^{\mathrm{PA}}(t_1, \ldots, \emptyset, \ldots, t_n) = \emptyset,$$

$$f^{\mathrm{PA}}(t_1, \ldots, \{\tau\}, \ldots, t_n) = \{\bar{\tau}_1, \ldots, \bar{\tau}_m\}, \quad \text{where } \tau \text{ is a } \cup\text{-free normal form and } \bar{\tau}_1 \cup \cdots \cup \bar{\tau}_m \text{ is the normal form of } \tau \text{ modulo } E,$$

$$f^{\mathrm{PA}}(t_1, \ldots, \{\tau_1, \ldots, \tau_m\}, \ldots, t_n) = \bigcup_{k=1}^{m} f^{\mathrm{PA}}(t_1, \ldots, \{\tau_k\}, \ldots, t_n);$$

- for $f: s_1 \times \cdots \times s_n \to s$ such that $s \notin S^{\cup}$ and $\forall j \in [1, \ldots, n]$, $s_j \notin S^{\cup}$, we have, if the terms $(t_j)_{1 \leq j \leq n}$ are normal forms,

$$f^{\mathrm{PA}}(t_1, \ldots, t_n) = t, \quad \text{where } t \text{ is the normal form of } f(t_1, \ldots, t_n).$$

In this definition, we used the fact that, for a term $t \in T^s_{S,\Sigma}$ with $s \notin S^{\cup}$ and for any t' such that $t \to^* t'$, we have that t' is $\cup$-free. Also, the cases (left open in the previous definition) where more than one sort of $S^{\cup}$ appear in the profile of an operator are handled similarly by distributivity. Lastly, though it is not conventional to represent the constants "$\emptyset_s$" by the empty set, it is considered here as a constant of PA^s, and no particular difficulty arises.

For instance, with Example 1.2, we have

$$\mathrm{eval}^{\mathrm{PA}}((a; b \,\|\, a; c)::x) = \{(a; a; b; c)::x, (a; a; c; b)::x, (a; b; a; c)::x, (a; c; a; b)::x\}.$$

We then obtain the following result.

Theorem 3.3. PA *is (isomorphic to) the initial model of the* $\cup$*-specification associated to* (R, E).

Proof. We have $E^{\mathrm{def}} = R_{\mathrm{L}}^{\mathrm{def}}$ and $E_{\cup} = E + R_{\mathrm{IND}}$. Using Lemma 3.2, we deduce that PA satisfies E^{def} and $E_{\cup}$. We simply need to establish that there exists a unique morphism from PA into the initial model $I = T_{S,\Sigma}/\equiv_{E^{\mathrm{def}} + E_{\cup}}$ of the $\cup$-specification. The only possible morphism φ is (denoting by $[t]_{E^{\mathrm{def}} + E_{\cup}}$ the class of a $\cup$-free term modulo $E^{\mathrm{def}} + E_{\cup}$):

- if $s \notin S^{\cup}$, then any $t \in \mathrm{PA}^s$ is a normal form in $T^s_{S,\Sigma}$ and $\varphi(t) = [t]_{E^{\mathrm{def}} + E_{\cup}}$;
- if $s \in S^{\cup}$ and t is a $\cup$-free normal form of $T^s_{S,\Sigma}$, then $\varphi(\{t\}) = [t]_{E^{\mathrm{def}} + E_{\cup}}$;
- if $s \in S^{\cup}$ and $(t_i)_{1 \leq i \leq n}$ are distinct $\cup$-free normal forms of $T^s_{S,\Sigma}$, then $\varphi(\{t_1, \ldots, t_n\}) = [t_1 \cup \cdots \cup t_n]_{E^{\mathrm{def}} + E_{\cup}}$;
- if $s \in S^{\cup}$, $\varphi(\emptyset) = [\emptyset_s]_{E^{\mathrm{def}} + E_{\cup}}$.

Then φ is well-defined; this is because if two *sets* of $\cup$-free normal forms of sort $s \in S^{\cup}$ are equal, then, by Lemma 3.2, the result of φ applied to either of these sets is identical. Lastly, case analysis shows that φ is a morphism. This terminates the proof of Theorem 3.3. □

The previous theorem ensures that, under the hypothesis of regularity, E-termination and E-Church-Rosserness, the intuition about $\cup$-specifications in terms of sets and union of sets is valid. In order to achieve these hypotheses, we now have the central result of this paper.

Theorem 3.4. *Let (R, E) be a regular $\cup$-system that is E-terminating. Then $\rightarrow_{R,E}$ is E-Church-Rosser iff $\mathrm{CP}(R_{\mathrm{L}}^{\mathrm{def}}, R_{\mathrm{L}}^{\mathrm{def}})$ is convergent.*

Proof. According to Section 2, and since NL is reduced to R_{I}, we need to check for the convergence of:

$$\mathrm{CP}(L, L)+\mathrm{CP}(L, E)+\mathrm{CP}(E, L)+\mathrm{ECP}(R_{\mathrm{I}}, R_{\mathrm{I}})+\mathrm{ECP}(R_{\mathrm{I}}, L)+\mathrm{ECP}(R_{\mathrm{I}}, E).$$

Let R_{ND} stand for $R_{\mathrm{N}}+R_{\mathrm{D}}$. The previous expression develops into

$$\mathrm{CP}(R_{\mathrm{L}}^{\mathrm{def}}, R_{\mathrm{L}}^{\mathrm{def}}) \tag{0}$$

$$+\mathrm{CP}(R_{\mathrm{ND}}, R_{\mathrm{ND}}) \tag{1}$$

$$+\mathrm{CP}(R_{\mathrm{ND}} \mathrm{m}\, R_{\mathrm{L}}^{\mathrm{def}})+\mathrm{CP}(R_{\mathrm{L}}^{\mathrm{def}}, R_{\mathrm{ND}}) \tag{2}$$

$$+\mathrm{CP}(R_{\mathrm{ND}}, E)+\mathrm{CP}(E, R_{\mathrm{ND}}) \tag{3}$$

$$+\mathrm{CP}(R_{\mathrm{L}}^{\mathrm{def}}, E)+\mathrm{CP}(E, R_{\mathrm{L}}^{\mathrm{def}}) \tag{4}$$

$$+\mathrm{ECP}(R_{\mathrm{I}}, R_{\mathrm{I}})+\mathrm{ECP}(R_{\mathrm{I}}, R_{\mathrm{ND}})+\mathrm{ECP}(R_{\mathrm{I}}, E) \tag{5}$$

$$+\mathrm{ECP}(R_{\mathrm{I}}, R_{\mathrm{L}}^{\mathrm{def}}). \tag{6}$$

Now we conclude:

- Expressions (1), (3) and (5) give rise to either empty or convergent critical sets. This comes from the fact that $\rightarrow_{\mathrm{IND,AC}}$ is Church-Rosser.
- Regarding expression (2), $\mathrm{CP}(R_{\mathrm{N}}, R_{\mathrm{L}}^{\mathrm{def}})$ is empty, because any left-hand side of $R_{\mathrm{L}}^{\mathrm{def}}$ is $\cup$-free. Thus, such a left-hand side only unifies with the left-hand side "$x \cup \emptyset$" of R_{N} at a variable occurrence. Suppose now that a left-hand side of a rule $\lambda \rightarrow \rho$ of $R_{\mathrm{L}}^{\mathrm{def}}$ unifies with (an occurrence of) the left-hand side "$f(x_1, \ldots, x_{i-1}, x \cup y, x_{i+1}, \ldots, x_n)$" of R_{D}. This means that λ is of the form "$f(e_1, \ldots, e_{i-1}, z, e_{i+1}, \ldots, e_n)$", due to the fact that λ is $\cup$-free. Then, $\mathrm{Var}(e_l) \cap \mathrm{Var}(e_m) = \emptyset$ if $l \neq m$, and there actually is a critical pair:

  $$\langle \rho\sigma, f(e_1, \ldots, e_{i-1}, x, e_{i+1}, \ldots, e_n) \cup f(e_1, \ldots, e_{i-1}, y, e_{i+1}, \ldots, e_n) \rangle,$$

  where $\sigma : x_i \backslash e_i,\ z \backslash x \cup y$. Now, $\rho\sigma \rightarrow_D^* \rho[z \leftarrow x] \cup \rho[z \leftarrow y]$. Since λ is linear, $\lambda \rightarrow \rho$ also applies to the second term of the critical pair, yielding $\rho[z \leftarrow x] \cup \rho[z \leftarrow y]$, too. The critical pair is convergent.

– The critical sets in expression (4) and (6) are empty.
Thus, expression (0) only, i.e., $CP(R_L^{def}, R_L^{def})$, may lead to nonconvergent critical pairs, whence the theorem. □

Example 3.5. The rules of Example 1.2 form a regular system that is E-terminating. We have $CP(R_L^{def}, R_L^{def}) = \emptyset$, which proves that the system is Church–Rosser.

Example 3.6. Consider the following set of rules:

$$\{f(0) \to 0, \quad f(s(x)) \to 2n(x) \cup f(x),$$
$$2n(0) \to 0, \quad 2n(s(x)) \to s(s(2n(x))),$$
$$\text{even?}(0) \to T, \quad \text{even?}(s(0)) \to F,$$
$$\text{even?}(s(s(x))) \to \text{even?}(x)\}.$$

Thus, $f(x)$ computes nondeterministically an even integer between 0 and $2x$. As previously, this system is regular, E-terminating, and $CP(R_L^{def}, R_L^{def}) = \emptyset$. It is therefore E-Church–Rosser.

The hypothesis of left-linearity in Theorem 3.4 is necessary, as shown with the following E-terminating, left-$\cup$-free but *not* left-linear system:

$$\{k \to a \cup b \quad g(x, x) \to x, \quad g(a, b) \to c, \quad g(b, a) \to c\}.$$

We have $CP(R_L^{def}, R_L^{def}) = \emptyset$. However,

$$g(k, k) \to k \to a \cup b, \qquad (E\text{-normal form})$$
$$g(k, k) \to g(a \cup b, a \cup b) \to g(a, a) \cup g(b, b) \cup g(a, b) \cup g(b, a) \to a \cup b \cup c \qquad (E\text{-normal form})$$

and the system is not E-confluent. Note that, actually, in this case, $CP(R_L^{def}, R_D)$ contains the nonconvergent critical pair "$\langle x \cup y, x \cup y \cup g(x, y) \cup g(y, x)\rangle$", from the term "$g(x \cup y, x \cup y)$". For such systems, one actually has the analogue of Theorem 3.4:

Proposition 3.7. *Let (R, E) be a left-$\cup$-free $\cup$-system that is E-terminating. Then, $\to_{R,E}$ is E-Church–Rosser iff $CP(R_L^{def}, R_L^{def}) + CP(R_D, R_L^{def}) + CP(R_L^{def}, R_D)$ is convergent.*

The proof is as for Theorem 3.4. This result applies to the last example, to prove that the corresponding system is not E-confluent.

3.1. Completion procedure

We can apply completion procedures (cf., e.g., [13, 24, 26, 27, 39]), in order to transform a system E^{def} describing a $\cup$-specification into a $\cup$-system. The results of [29] for instance, immediately apply to the case where one would accept any

kind of $\cup$-system (not necessarily regular) as an output. However, since we are mainly interested in regular systems, let us simply recall what a completion procedure may be in the regular case.

We recall that an *E-reduction ordering* is a quasi-ordering ">" on terms that is compatible with the operators of Σ such that the associated equivalence relation $t > t' \,\&\, t' > t$ contains $\equiv_E$, and such that the associated ordering $t > t' \,\&\, \neg(t' > t)$ is well-founded.

Procedure COMPLETE(R, Eq, n)
/* Initial call to the procedure is COMPLETE($\emptyset$, E^{def}, 0). ">" is a given E-reduction ordering */
  while Eq is not empty **do** {
    choose an equation $\lambda = \rho$ in Eq, and remove it from Eq;
    Compute $\lambda\downarrow$ and $\rho\downarrow$, the normal forms for $\rightarrow_{R,E}$ of λ and ρ;
    case
      - $\lambda\downarrow \equiv_E \rho\downarrow$ **then** COMPLETE(R, E^{def}, Eq)
      - $\lambda\downarrow > \rho\downarrow$ and $R + \{\lambda \rightarrow \rho\}$ is regular **then** SIMPLIFY(R, Eq) BY "$\lambda \rightarrow \rho$"
      - $\rho\downarrow > \lambda\downarrow$ and $R + \{\rho \rightarrow \lambda\}$ is regular **then** SIMPLIFY(R, Eq) BY "$\rho \rightarrow \lambda$"
      - **else** STOP-with-FAILURE}
  if all rules in R are marked **then** STOP-with-SUCCESS
  else
    choose *fairly* an unmarked rule $\lambda \rightarrow \rho$ in R and assign label "n" to it;
    let CP stand for the set of critical pairs computed between $\lambda \rightarrow \rho$ and the rules of R of label smaller than or equal to "n";
    Eq $\leftarrow$ Eq + CP;
    mark the rule $\lambda \rightarrow \rho$ in R;
    COMPLETE(R, E, $n+1$)
end-procedure COMPLETE

with:

Procedure SIMPLIFY(R, Eq) BY "$\lambda \rightarrow \rho$"
  Let K be the set of the labels of the rules of R the left-hand side l_k of which is reducible by $\lambda \rightarrow \rho$, say into l'_k;
  $R \leftarrow \{l_k \rightarrow r'_k \mid l_k \rightarrow r_k \in R,\ k \notin K$ and r'_k is a normal form of r_k for $\rightarrow_{R \cup \{\lambda \rightarrow \rho\}, E}\} + \{\lambda \rightarrow \rho\}$
  Eq $\leftarrow$ Eq + $\{l'_k = r_k \mid l_k \rightarrow r_k \in R$ and $k \in K\}$
end-procedure SIMPLIFY_BY_

Choosing "*fairly*" the rules in the procedure COMPLETE means that no rule is infinitely often choosable without being chosen eventually. We then have the following result.

Theorem 3.8. *When the completion procedure stops with success, the resulting set of rules R is regular, E-terminating and E-Church–Rosser, and* $\equiv_{E+E^{\text{def}}} = \equiv_{E+R}$. *Thus,* $t \equiv_{E+E^{\text{def}}} t'$ *iff* $t\downarrow_{R,E} \equiv_E t\downarrow_{R,E}$.

Note. Suppose, for instance, that in the "**case**" statement of the completion procedure, it happens that $l\downarrow > r\downarrow$, but $R+\{l \to r\}$ is *not* regular (i.e., "l" is not linear or $\cup$-free). It is possible, however, to carry on the completion with the *general* completion procedure *modulo* E; in case of success of the generalized completion, the resulting system may of course not be regular.

Example 3.9. Let us add to the (complete) system of Example 3.6 the following rule:

$$\text{even?}(f(x)) \to \text{T}.$$

The completion procedure applied to the whole system runs as follows:
- there is a critical pair between the first and the last rule: $\langle \text{even?}(0), \text{T}\rangle$, which trivializes into $\langle \text{T}, \text{T}\rangle$;
- there is a critical pair between the second and the last rule: $\langle \text{even?}(2n(x) \cup f(x)), \text{T}\rangle$, which normalizes into $\langle \text{even?}(2n(x)) \cup \text{T}, \text{T}\rangle$. The rule

  $$\text{even?}(2n(x)) \cup \text{T} \to \text{T}$$

  is added to the system, which is not regular anymore. The general completion procedure has to start on the current state of the system;
- there is a critical pair between this last rule and the third rule: $\langle \text{even?}(0) \cup \text{T}, \text{T}\rangle$, which trivializes into $\langle \text{T}, \text{T}\rangle$;
- there is a critical pair between this last rule and the fourth rule: $\langle \text{even?}(\text{s}(\text{s}(2n(x))) \cup \text{T}, \text{T}\rangle$, which trivializes into $\langle \text{T}, \text{T}\rangle$.

The procedure stops, and the resulting system is complete, though not regular. We remark that, in this case, there exists a regular presentation of the same specification, namely replacing the last rule by the rule: $\text{even?}(2n(x)) \to \text{T}$.

Lastly, note that, as shown in the next section, the previous script may be read as a *proof* that "$\text{even?}(f(x)) = \text{T}$" is an inductive theorem of the theory.

4. Proofs in $\cup$-specifications

In this section, we address the question of the proof of inductive properties, via so-called inductionless induction methods (cf. [13, 18, 23, 27, 28, 34]). We recall the following definition.

Definition 4.1. A property $t = t'$ is an *inductive theorem* of a specification iff, for any good substitution σ, $t\sigma$ and $t'\sigma$ are congruent via the axioms of the specification.

We choose to present the inductionless induction methods in the flavor of the recent work of [28] that rely on the notion of *inductive reducibility*, and which is

simpler in its principle than previous approaches. However, this work is still under development; in particular, it is still not established whether inductive reducibility with nonlinear rules (such as R_I, and not to mention R_{NL}^{def} that may be supposed empty here) is decidable when dealing with AC operators, though it is likely to be so. Also, a drawback is that, so far, there exist only prototypes of implementations for this approach; and actually, the experiments accounted here have been performed on systems that do not encompass these new features.

We suppose here that a regular, E-terminating and E-Church-Rosser $\cup$-system is given. We have the following definition.

Definition 4.2. (1) A term is *E-inductively reducible* iff all its ground instances are reducible for $\rightarrow_{R/E}$.

(2) A term is *freely inductively reducible* iff all its ground, $\cup$-free instances are reducible for $\rightarrow_{R_L^{def}}$.

We have the following lemma.

Lemma 4.3. *A $\cup$-free term is E-inductively reducible iff it is freely inductively reducible.*

Proof. Suppose that t is E-inductively reducible. Let σ be a ground, $\cup$-free substitution. $t\sigma$ is reducible by $\rightarrow_{R/E}$, and thus necessarily by $\rightarrow_{R_L^{def}}$ since no equation of R_{IND} can be applied (modulo E). This means that t is freely inductively reducible;

Conversely, suppose that t is freely inductively reducible. Let σ be a ground substitution. If σ is $\cup$-free, then $t\sigma$ is $\rightarrow_{R_L^{def}}$-reducible. Otherwise, since t admits a root symbol and $t\sigma$ contains at least a "$\cup$" symbol (or a "$\emptyset$" symbol), the rule R_D (or R_N) applies to $t\sigma$. In any case, $t\sigma$ is $\rightarrow_{R/E}$-reducible. □

This allows to extend the results of [28] to our framework. To this effect, the completion procedure of Section 3.2 is modified in the following fashion. The statement SIMPLIFY(R, Eq) BY "$\lambda \rightarrow \rho$" is replaced by

> **if** λ is not freely inductively reducible by R_0
> **then** STOP-with-DISPROOF
> **else** SIMPLIFY(R, Eq) BY "$\lambda \rightarrow \rho$".

The statement "SIMPLIFY(R, Eq) BY '$\rho \rightarrow \lambda$'" is modified similarly. The resulting procedure IND-COMPLETE is called *inductive completion procedure.*

We now suppose that a regular, E-terminating and E-Church-Rosser $\cup$-system R_0 is given and that one wishes to determine if a set of equations Eq is inductively valid w.r.t. $\equiv_{E+R_0}$. The procedure IND-COMPLETE will be run with the initial call "IND-COMPLETE(R_0, Eq, 0)". Note that the formal parameter "R" is initially bound to "R_0", and that its value will change. On the other hand, "R_0" is also used as a constant throughout the procedure (in the modified statement hereabove). We then have the following result.

Theorem 4.4. *If* IND-COMPLETE *runs without reaching the* "STOP-with-FAILURE" *statement, then*:

(1) Eq *is inductively valid iff the procedure reaches the* "STOP-with-SUCCESS" *statement, or runs forever*;

(2) Eq *is not inductively valid iff the procedure reaches the* "STOP-with-DISPROOF" *statement.*

Proof. We notice that in the modified statement hereabove, if $R+\{\lambda \to \rho\}$ is regular, then λ is $\cup$-free. Thus, in order to check that λ is E-inductively reducible, it is enough to check that it is freely inductively reducible, according to Lemma 4.3. Now, except for this difference, our procedure IND-COMPLETE and the one of [28] are similar, and the validity results proven there also apply to our case. □

Example 4.5. The system given in Example 1.2 is regular, E-terminating and E-Church-Rosser. The inductive completion procedure is applied to the formula

$$((p \mathbin{\lfloor\!\lfloor} q) \mathbin{\lfloor\!\lfloor} r)::x = (p \mathbin{\lfloor\!\lfloor} (q \parallel r))::x$$

that normalizes into

$$((p \mathbin{\lfloor\!\lfloor} q) \mathbin{\lfloor\!\lfloor} r)::x = (p \mathbin{\lfloor\!\lfloor} (q \mathbin{\lfloor\!\lfloor} r) + (r \mathbin{\lfloor\!\lfloor} q))::x.$$

With our completion procedure, there is *only* one critical pair to consider,[2] from the term "$(((a;p) \mathbin{\lfloor\!\lfloor} q) \mathbin{\lfloor\!\lfloor} r)::x$", that becomes trivial after normalization. Free inductive reducibility is checked by hand; this is easy in this case since "$\parallel$", "$\mathbin{\lfloor\!\lfloor}$" and "::" have *complete* definitions. The previous formula is therefore an inductive theorem of the specification. □

It is interesting to notice that the equation

$$((p \parallel q) \parallel r)::x = (p \parallel (q \parallel r))::x$$

is now an equational theorem of the completed system.

This example has also been treated in REVE-2 (cf., e.g., [34, 37]), i.e., with a general AC completion procedure. The system had to compute *111* critical pairs (which all became trivial), to be compared with the *one* critical pair computed with our approach. This clearly demonstrates the gain that we obtain for regular $\cup$-theories.

These kind of specifications for concurrent systems have also been considered, for instance in [4, 5, 6]. There too, *all* possible critical pairs had to be computed in an *ad hoc* fashion [6]. Thus, when compared with related works, our formalism provides a general framework for the proof of such properties, in a quite efficient way.

Lastly, as in the previous section, notice that if the current system is not regular any longer at a given step of the completion (either because an equation to be proven could not be oriented into a left-$\cup$-free, left-linear rule, or because the procedure produced itself such a kind of equation), then it may still be possible to apply the *general* inductive completion procedure of [26, 28].

[2] Taking into account only *one* axiom such as $(a;p) \mathbin{\lfloor\!\lfloor} q = a;(p \parallel q)$.

Example 4.6. Still with the same specification, the formula

$$((p;q);r)::x = (p;(q;r))::x$$

is an *equational* theorem of the specification. Consider the more complicated equation

$$(((p;q);r)\|s)::x = ((p;(q;r))\|s)::x$$

which is not an equational theorem. After normalization, both sides contain "$\cup$" symbols, and the corresponding system cannot be turned into a regular one. We therefore made the experiment of running the *general* inductive completion procedure on this example, under the system REVE-2. Three new rules

$$((\delta;p)\|\!\!\lfloor q)::x \to \emptyset,$$

$$(((p;q);r)\|\!\!\lfloor s)::x \to ((p;(q;r))\|\!\!\lfloor s)::x,$$

$$(s\|\!\!\lfloor((p;q);r))::x \to (s\|\!\!\lfloor(p;(q;r)))::x$$

were generated, and the procedure stopped, thus proving that the previous formula was an inductive theorem of the specification.

However, in this general case, the amount of computation (*modulo* AC) is far larger than in the case covered by Theorem 4.4, as shown above.

5. Conclusion

In this paper, we have considered the feasibility of term-rewriting systems taking into account nondeterminism. Such systems specify sets of possible results, and choice is simulated by the union of such sets. The main characteristic of this work is that it leads to *automatic* theorem-proving methods, extending to nondeterministic specifications principles now widely used in the deterministic framework and implemented in large systems. The class of the regular systems that correspond to natural restrictions on the specification of the choice may be treated in a simple way, with little use of AC-pattern-matching or unification procedures. For these systems, the completion procedures that we present are far more efficient than the classical AC completion procedures. We have developed several examples illustrating the applicability of the method to nontrivial specifications.

The main limitation of general term-rewriting systems, considering that the work accounted here provides an alternative to the confluence requirement, remains now that nonterminating computations still cannot be taken into consideration. A possible solution would be to consider only their finite approximations. For instance, in order to specify a process such that $P(x) \equiv a(x);P(x)$, we can synthesize the process $\bar{P}(x, n)$ defined (in a finitely terminating fashion) by

$$\bar{P}(x, 0) = \delta \quad \text{and} \quad \bar{P}(x, s(n)) = a(x);\bar{P}(x, n),$$

and prove properties about $\bar{P}(x, n)$ for every n. But this will not allow to deal with properties such as fairness or partial correctness. The work of [41] may be relevant w.r.t. such issues.

Acknowledgment

I wish to thank J.-P. Jouannaud, M.-C. Gaudel, G. Bernot, C. Choppy and F. Voisin for fruitful discussions and suggestions.

This work has been partially supported by the Esprit METEOR project and the P.R.C. de Programmation of the CNRS. It has been partially realized while the author was visiting the Weizmann Institute (Rehovot, Israel), with support of the INRIA.

References

[1] J.A. Goguen, J.W. Thatcher and E.G. Wagner, An initial algebra approach to the specification, correctness and implementation of abstract data types, in: *Current Trends in Programming Methodology* (Prentice-Hall, Englewood Cliffs, NJ, 1978).

[2] K. Apt, Ten years of Hoare's logic: a survey. Part II: nondeterminism, *Theoret. Comput. Sci.* **28** (1984) 83-109.

[3] L. Bachmair and D. Plaisted, Associative path orderings, in: *Proc. 1st RTA Conf.*, Dijon (1985) 241-256.

[4] J. Bergstra and J. Klop, Algebra of communicating processes with abstraction, CWI Report CS-R8403, Amsterdam (1984).

[5] J. Bergstra and J. Klop, Algebra of communicating processes. Part II, CWI Report, Amsterdam (1984).

[6] J. Bergstra and J. Klop, Syntax and defining equations for an interrupt mechanism to process algebra, CWI Report CS-R8503, Amsterdam (1985).

[7] M. Bidoit, Une méthode de présentation de types abstraits: applications, Thèse de 3ème cycle, Orsay (1981).

[8] R. Boyer and J.S. Moore, *A Computational Logic* (Academic Press, New York, 1979).

[9] G. Boudol, An "asynchronous" calculus MEIJE, in: *NATO Summer School*, La-Colle-sur-Loup, France (1984).

[10] S.D. Brookes, On the relationship between CCS and CSP, in: *Proc. 10th ICALP*, Lecture Notes in Computer Science **154** (Springer, Berlin, 1983).

[11] M. Broy, On the Herbrand Kleene universe for nondeterministic computations, in: *Proc. MFCS'84 Conf.*, Lecture Notes in Computer Science **176** (Springer, Berlin, 1984).

[12] M. Broy and M. Wirsing, On the algebraic specification of nondeterministic programming languages, in: *Proc. CAAP'81*, Lecture Notes in Computer Science **112** (Springer, Berlin, 1981).

[13] B. Buchberger, History and basic features of the critical-pair/completion approach, in: *Proc. 1st RTA Conf.*, Lecture Notes in Computer Science **202** (Springer, Berlin, 1985) 1-45.

[14] C. Choppy and C. Johnen, Petrireve: proving Petri net properties with rewriting systems, in: *Proc. 1st RTA Conf.*, Lecture Notes in Computer Science **202** (Springer, Berlin, 1985) 271-286.

[15] N. Dershowitz, Orderings for term rewriting systems, in: *Proc. 20th Symp. on Foundation of Computer Science* (1979) 123-131.

[16] D. Detlefs and R. Forgaard, A procedure for automatically proving the termination of a set of rewrite rules, in: *Proc. 1st RTA Conf.*, Lecture Notes in Computer Science **202** (Springer, Berlin, 1985) 255-270.

[17] L. Fribourg, A narrowing procedure for theories with constructors, in: *Proc. CADE Conf.*, Napa (1984) 259-281.

[18] J. Goguen and H. Meseguer, Completeness of many-sorted equational logic, *SIGPLAN Notices* (1981).

[19] J. Goguen, How to prove algebraic inductive hypotheses without induction, in: *Proc. 5th CAD*, Les Arcs, France (1980) 356-373.

[20] M. Hennessy, Powerdomains and nondeterministic recursive definitions, in: *Proc. Symp. on Programming*, Lecture Notes in Computer Science **137** (Springer, Berlin, 1982).

[21] J. Hsiang and M. Rusinowitch, On word problems in equational theories, submitted for publication (1987).

[22] G. Huet, Confluent reductions: abstract properties and applications to term rewriting systems, in: *Proc. 18th FOCS*, Providence, RI (1977) 30-45.

[23] G. Huet and J.-M. Hullot, Proofs by induction in equational theories with constructions, in: *Proc. 21st FOCS* (1980).

[24] G. Huet and D.C. Oppen, Equations and rewrite rules: a survey, in: R. Book, ed., *Formal Languages: Perspective and Open Problems* (Academic Press, New York, 1980) 349-305.

[25] C.A.R. Hoare, Communicating sequential processes, *Comm. ACM* **21** (1978) 666-677.

[26] J.-P. Jouannaud and H. Kirchner, Completion of a set of rules modulo a set of equations, in: *Proc. 11th POPL* (1984) 83-92.

[27] J.-P. Jouannaud, in: J. van Leeuwen, ed., *Handbook in Theoretical Computer Science*, to appear (1988).

[28] J.-P. Jouannaud and E. Kounalis, Automatic proofs by induction in theories without constructors, L.R.I. Technical Report, University of Paris-South (1987).

[29] S. Kaplan, Rewriting with a nondeterministic choice operator: from algebra to proofs, in: *Proc. 1st ESOP*, Lecture Notes in Computer Science **213** (Springer, Berlin, 1982).

[30] S. Kaplan, Simplifying conditional term rewriting systems, *J. Symbolic Comput.*, to appear (1988).

[31] S. Kaplan and A. Pnueli, Specification and implementation of concurrently accessed data structures, in: *Proc. 4th STACS*, Lecture Notes in Computer Science **247** (Springer, Berlin, 1987) 220-244.

[32] C. Kirchner, A new equational unification method: a generalization of Martelli-Montanari's algorithm, in: *Proc. CADE Conf.* (1984).

[33] C. Kirchner, Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles, Thèse d'Etat, University of Nancy (1985).

[34] C. Kirchner and H. Kirchner, REVEUR-3, *Sci. Comput. Programming* **8** (1987) 69-86.

[35] D.E. Knuth and P.B. Bendix, Simple word problems in universal algebra, in: J. Leech, ed., *Computational Problems in Abstract Algebra* (Pergamon, Oxford, 1970).

[36] E. Kounalis, Validation de spécifications algébriques par complétion inductive, Thèse d'Etat, University of Nancy, France (1985).

[37] P. Lescanne, Computer experiment with the REVE term rewriting systems generator, in: *Proc. 10th POPL Conf.* (1983) 98-108.

[38] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science **92** (Springer, Berlin, 1980).

[39] D.L. Musser, On proving inductive properties of abstract data types, in: *Proc. 7th POPL Conf.*, Las Vegas (1980) 154-162.

[40] M. Nivat, Nondeterministic programs: an algebraic overview, in: *Proc. IFIP'80* (North-Holland, Amsterdam, 1980) 17-28.

[41] S. Porat and N. Francez, Fairness in term rewriting systems, in: *Proc. 1st RTA Conf.*, Lecture Notes in Computer Science **202** (Springer, Berlin, 1985).

[42] G. Peterson and M. Stickel, Complete sets of reductions for some equational theories, *J. ACM* **28** (1981) 233-264.

[43] A. Poigné, On effective computations of nondeterministic schemes, in: *Proc. CAAP'81*, Lecture Notes in Computer Science **112** (1981).

[44] W. Reisig, *A Petri Net Primer* (Springer, Berlin, 1985).

[45] W. Reisig, On the semantics of Petri nets, in: E.J. Neuhold and G. Chroust, eds., *Formal Models in Programming* (North-Holland, Amsterdam, 1985).

[46] J.L. Rémy and H. Zhang, Contextual rewriting, in: *Proc. 1st RTA Conf.*, Lecture Notes in Computer Science **202** (Springer, Berlin, 1985) 46-62.